공학석사학위논문

레이저 스캐너 기반 자율주행용 교차로 내 주변 차량 인지 알고리즘 개발

Surrounding Vehicle Perception Algorithm in Intersection for Autonomous Vehicle with Laser scanner

2017 년 8 월

서울대학교 대학원

기계항공공학부

김 선 욱

Abstract

Surrounding Vehicle Perception Algorithm in Intersection for Autonomous Vehicle with Laser scanner

Seonwook Kim
School of Mechanical and Aerospace Engineering
The Graduate School
Seoul National University

The aim of this study is designing surrounding vehicle movement perception algorithm in urban condition. In recent autonomous vehicle industry, many researchers are focusing on three major topics which is called environment perception, localization and designing controller for autonomous vehicle these days. Especially for the perception technology, the design of the algorithm follows the characteristics of the target environment or objects. In Urban condition, to design safe drivable path for autonomous vehicle, the objects' position is much important than high way conditions. Also, various objects appear which cannot be detected with RADAR or yield fault case with camera. Because of these reasons, many research are trying to fuse different sensors including camera, RADAR and LiDAR to overcome the challenges that can occur in urban conditions, therefore, laser scanner based target detecting technology is needed to perceive in city road.

The tracking filter consists of two parts, shape estimation and tracking filter. To fuse with other sensors or designing target filter, there should be a step for compressing point cloud group information into some representative point or state. Thus in shape estimation parts, we transform the laser scanner's point cloud data into vehicle position state measurement value. Vehicle shape estimation also consists of two parts, clustering and shape extraction. Clustering classify the total point cloud into object level and shape extraction estimates the vehicle liked objects' position information. The clustering part works based on Euclidean

Minimum Spanning Tree (EMST), and for the shape extraction, Random Sample Consensus (RANSAC) method is used to estimate the target objects rear and side edge. The second part, tracking filter, has two different filters. Particle filter estimates the target vehicle's position including heading angle. To improve the tracking performance of the particle filter, Kalman filter is also designed to estimate the velocity and yaw rate recursively to update the process model of the particle filter.

The performance of the proposed algorithm has been verified with several stages. To check quantitative error level, off-line simulation is held for profile based motion tracking case and designed intersection simulator with simple path tracking algorithm for the target vehicle. In these conditions, the exact target vehicle's position information was known, thus we verified the error level of the lateral/longitudinal direction of target vehicle's local coordinate which is important information when designing driving path or controller. For the second step, simulation with point cloud data which is collected from the test vehicle was held to verify its performance for actual vehicle condition. As a final stage, for integrating into autonomous vehicle, the proposed algorithm evaluated into the test vehicle for guaranteeing on-line performance.

Keywords: Vehicle Perception Filter, Point Cloud Post-processing, Particle Filter, Autonomous Vehicle, Laser Scanner, Recursive Structure

Student Number: 2015-20716

Contents

List of Figures

Nomenclatures

d_{ij}	: distance between i-th point & j-th point
ε	:threshold of clustering function
x_k	: estimated position state at time step k
v_k	: velocity at time step k
$\triangle t$	: unit time for discritization
l	: wheel-base of the vehicle
ϕ_k	: tire steering angle of the target vehicle at time step k
$p_{k,i}$	: i-th particle state at time step k
$n_{p,i}$	: noise for particle generation for i-th particle
a_k	: estimated input values at time step k
z_{k+1}	: measurement info(result of shape extraction) at time step k
$w_{k+1,i}$	: updated weight of i-th particle for time step k+1
n	: number of particle
$p'_{k+1,i}$	: resampled i-th particle for time step k+1
$w'_{k+1,i}$	: resampled i-th particle's weight for time step k+1
$x_{vel,k}$	:state vector for velocity estimating Kalman filter at time step k
$x_{steer,k}$	: state vector for yawrate estimating Kalman filter at time step k
$z_{pos,k}$	: position information from particle filter at time step k
$z_{yaw,k}$	: yaw angle information from particle filter at time step k
$z_{vel,k}$	: estimated measurement of velocity filter at time step k
$z_{steer,k}$	: estimated measurement of yawrate filter at time step k

$\overline{x}_{vel,k}$: process updated state of velocity filter at time step k

$\overline{x}_{steer,k}$: process updated state of yawrate filter at time step k

$\hat{x}_{vel,k}$: measurement updated state of velocity filter at time step k

$\hat{x}_{steer,k}$: measurement updated state of yawrate filter at time step k

H_{vel} : observation matrix of velocity Kalman filter

H_{steer} : observation matrix of yawrate Kalman filter

A_{vel} : process update matrix of velocity Kalman filter

A_{steer} : process update matrix of yawrate Kalman filter

Q_{vel} : process noise matrix of velocity Kalman filter

Q_{steer} : process noise matrix of yawrate Kalman filter

V_{vel} : measurement noise matrix of velocity Kalman filter

V_{steer} : measurement noise matrix of yawrate Kalman filter

$M_{vel,k}$: covariance matrix of $\overline{x}_{vel,k}$

$M_{steer,k}$: covariance matrix of $\overline{x}_{steer,k}$

$P_{vel,k}$: covariance matrix of $\hat{x}_{vel,k}$

$P_{steer,k}$: covariance matrix of $\hat{x}_{steer,k}$

$v_{action,k+1}$: estimated target vehicle's velocity at time step k+1

$\dot{\psi}_{action,k+1}$: estimated target vehicle's yawrate at time step k+1

Chapter 1

Introduction

1.1 Research Background

Recently, the interest of automotive researches moves from the passive safety system to the active safety system and at the same time, researches about sensors or its processing technologies for automated driving system or ADAS system is also actively held [Enache10],[Hoedemaeker98],[O'Malley10]. Furthermore, there are various ongoing projects about the safety technology in automated driving system. There is an ongoing project called 'Vision Zero' in Sweden of which objective is to reduce fatalities to zero by 2020 [Tingvall00]. The research initiative PReVENT is one step closer to market-ready automated vehicles. In this project, low-cost sensors, for instance, cameras or radio-based car to car communication were used [Schulze05]. Also, several researches keep focusing on advancing and developing the autonomous driving technology.

Still, most of the commercialized ADAS system and automated driving system mainly concentrate on the high way conditions. The urban road has huge difference in the motion of the surrounding environments and their road curvature characteristics. In city road, turning motion, which has frequently

occurs at the intersection and stop & go situation which has more frequently occur in traffic jam. In these situation, the surrounding objects such as other vehicles and pedestrians acts with more nonlinear and random motions. Especially for the other vehicles, their various heading angle changes makes difficulty to predict their future motion and intention or yield fault detecting cases. Concentrating on intersection, one of the most common road condition in city road, the issues of nonlinear & random movement of surroundings become more important. In intersection, other vehicles can choose their direction freely, for instance, left or right turn and even U-shape turn. Because of these high degree of freedom, the tracking filter should track these motions faster. At the same time, shape of the intersection is hard to determine as a specific size because there are various type of them. Also, designing for typical cross road shape might occur extra issues when evaluating to different shape, such as roundabout. For these reasons, in this research, we focus on designing a surrounding vehicle position tracking filter that can detect various direction change and random motion with recursive structure of particle filter and Kalman filter based on laser scanner data.

1.2 Research Overview

The algorithm is composed with two functional parts, clustering & shape extraction and vehicle tracking particle filter. The clustering & shape extraction algorithm is design with Euclidean minimum spanning tree (EMST) based data classification and estimates the target vehicle's rear center position information with Random Sample Consensus (RANSAC) method. After the measurement data from the sensor transfer into the filter's state vector form, the particle filter works to estimate the vehicle's position. To overcome the difference of the filter's process model and the actual target vehicle's motion and reflect the motion change, Kalman filter is also designed to compensate the difference and update the process model of the tracking filter recursively.

The performance of the proposed perception algorithm is evaluated with four different stages. To compute exact tracking performance, knowing the all the surrounding target vehicles' actual position in experiment is quite tough, simulation level verification was held first in section 3. Simulated with predefined velocity and yaw rate and checked the performance. As a second step, simulation using intersection simulator was held because the main target of this study is intersection.

Finally using test vehicles, we verified its performance stating vehicle experiment data based simulation and actual test vehicle evaluation at the intersection and several urban road conditions.

Chapter 2

Target Vehicle Tracking Filter

The overall system architecture is shown in Figure 2.1. The laser scanner sensor gives point cloud data and the final goal of the algorithm is to obtain surrounding vehicles position and heading angle. And the further purpose of the algorithm is to estimate the current state of the vehicles and predict their intention and give these information to ego-vehicle planner or controller to avoid collision or realize ego-vehicle's current driving situation. The algorithm is composed with two parts, clustering & shape extraction part and vehicle tracking filter part. Clustering & shape extraction part computes the estimated state vector measurement from processing sensor data, in our case, laser scanner point cloud data. The vehicle tracking filter estimates the position of the vehicle based on the measurement and the simple kinematic process model. To improve the tracking performance, the particle filter's process model is updated with the estimated input from Kalman Filter.

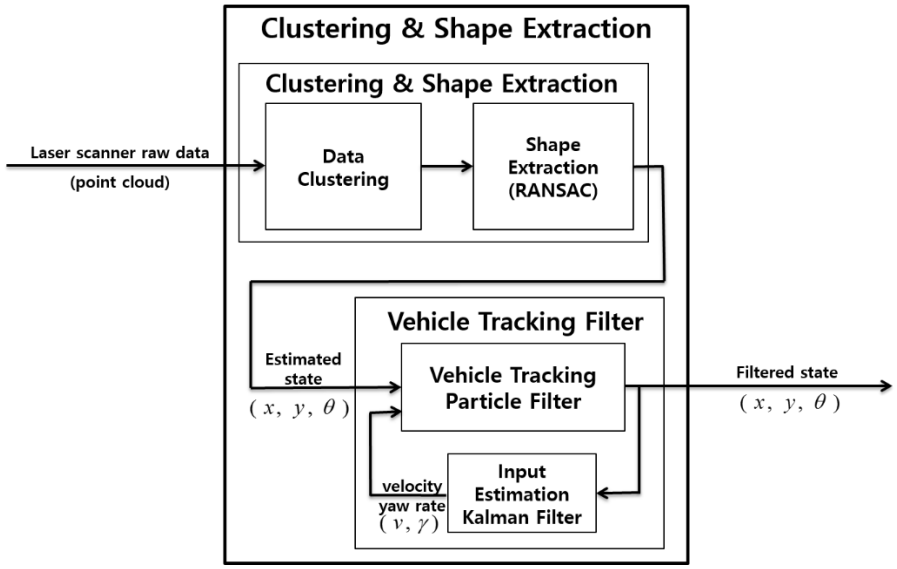


Figure 2.1. Overall Architecture of Vehicle Tracking Filter

2.1 Laser scanner Data Post-processing

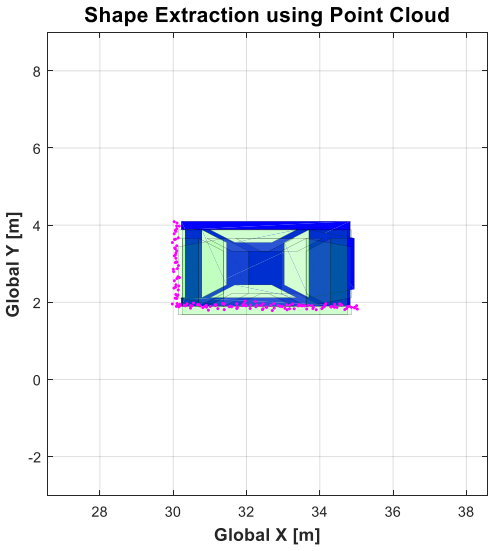
To design surrounding vehicle tracking filter, process of obtaining the filter's state variable from the sensor raw data. In our research, we assumed that the ego vehicle is equipped with laser scanners to detect intersections. While using laser scanner sensors, the raw data are given as point cloud data. Also in our research, the particle filter is designed to track surrounding vehicles and the state variable of the filter is the vehicles' rear center position and heading information, $(x,\ y,\ \psi)$. To construct the position data from the point cloud data, we design algorithm which is called clustering & shape extraction to estimate the rear center of the vehicle and generate the measurement for the filter.

In the clustering part, the algorithm separates the whole point data into each surround targets. As mentioned before, two one-layer LiDAR sensors model is assumed for surrounding perception in our system. For the real time performance, we used the on grid data instead of the raw data points. We decide the lateral and longitudinal grid size as 5 cm, 5 cm each. For the clustering process, we used EMST method introduced in [Wang12]. To improve the computational load of the algorithm, we used the distance thresholding function to transfer the Euclidean distances into simple logical numbers [Jeong16]. In this study, $\varepsilon = 0.5m$ is used. The distance function is as follows:

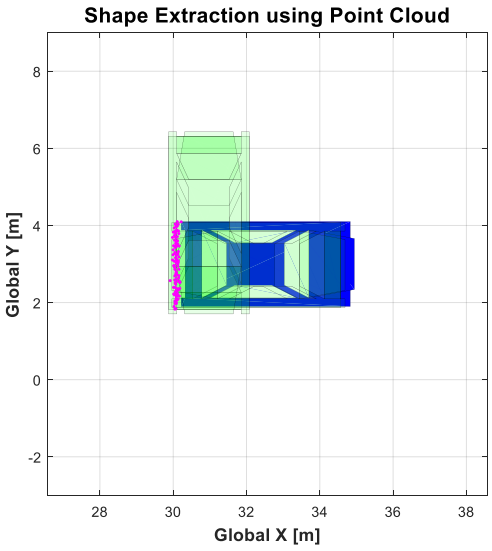
$$f\left(d_{ij}\right) = \begin{cases} 1 & d_{ij} \le \varepsilon \\ 0 & d_{ij} > \varepsilon \end{cases} \tag{2.1}$$

As a next step, to estimate the rear center position, Random Sample Consensus

(RANSAC) algorithm is used [Derpanis10]. Usually when detect the vehicles with laser scanner, especially in one layer condition, the data shape forms like I shape or L shape [Kaempchen04]. Thus the vehicles data from the laser scanner can be classified by the line elements. Also, the line element of the vehicle data gives the heading direction of the vehicle. To estimate the vehicles rear center position, we assumed that the surrounding vehicles have their geometry with length 5m & width 2.2m. RANSAC algorithm select number of points randomly, and compute the line as 1st order polynomial. After obtaining the equation of line, decide the outlier of the estimation. Iteratively computing this ratio of outlier value for several random sampling, choose the best estimation of the line to choose as the side or rear line of the vehicle. After the line estimation process is over, by using principle axis transformation to estimate the center position and heading angle. In choosing the heading angle, especially for the I shape, we assumed that the vehicle can be go backward and the detected line might be either side or rear line of the object, thus for the same center position, we generated two heading directions for each, which has exactly the opposite direction. As the tracking filter is using rear center information, coordinate transformation is once held, and give the rear center position information and yaw angle of the target object to the tracking filter. The results of the shape extraction from the point clouds is shown in figure 2.2.(a)-(b) each.



(a)



(b)

Figure 2.2. Vehicle Shape Extracting results with Laser scanner

(a) L shape case (b) I shape case

2.2 Vehicle Tracking with Particle Filter

After the rear position is estimated from the post- processing, particle, we designed filter to track the vehicles current position. Still, for perceiving in intersection or parking lot, as mentioned above, vehicles changes their heading angle frequently. At the same time, the dynamics of the vehicles are defined by the cosine and sine function of heading angle which makes the nonlinearity of the motion and cause disadvantages for the linear filter to track them. For the better performance, in our research, the particle filter is designed to track the surroundings position. The overall process of the particle filter algorithm is shown in Figure 2.3.

For the process model of the filter, simple car model is considered. As mentioned above, the state variables of the filter is vehicle's rear center position. The exact dynamic model equations are as follows in figure 3.

$$x_{k+1} = x_k + \begin{bmatrix} v_k \cos\psi \\ v_k \sin\psi \\ v_k / l \cdot \tan\phi_k \end{bmatrix} \Delta t \qquad (2.2)$$

$$where\ x_k = \begin{bmatrix} X & Y & \psi \end{bmatrix}$$

Where v_k is the velocity of the target vehicle and l is the wheel-base of the vehicle. Finally, ϕ_k is the steering angle of the vehicle. In the equation, the v_k and ϕ_k are the input of the filter. Still, the 3rd row of the equation shows that yaw rate element is the combination of two independent inputs. Thus we consider the model has two inputs of $v_k, \dot{\psi}$ instead of v_k, ϕ_k. The reason for considering different inputs because the 3rd row's equation has function tangent, which also gives nonlinearity to the system. The final transferred the dynamic equation into separated form of the inputs

and state become as follows.

$$x_{k+1} = x_k + \Delta t \begin{bmatrix} \cos\psi & 0 \\ \sin\psi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ \dot{\psi}_k \end{bmatrix} \tag{2.3}$$

$$where\ x_k = \begin{bmatrix} X & Y & \psi \end{bmatrix}$$

As a likelihood function to define the weight of each particle and for particle generation noise model, the Gaussian normal distribution is used. The measurement of the filter is used the previous estimated rear center position.

Algorithm 1. Particle Filter with Action Update

$p_{k,i}$:= i-th particle at time step k

x_k := Estimated rear center position at time step k

$n_{p,i}$:= Generated noise for particle generating

a_k := Input for process update at time step k

z_{k+1} := measurement at time step k

$w_{k+1,i}$:= weight of i-th particle at time step k

For each step,

for $i = 1$ to n

$p_{k,i} = x_k + n_{p,i}$ particle generation

$p_{k+1,i} = f(p_{k,i}, a_k)$ process update

$w_{k+1,i} = g(p_{k+1,i}, z_{k+1})$ compute weight

end

$w'_{k+1,i} = w_{k+1,i} / sum(w_{k+1,})$ weight *nomalize*

resample particle $p'_{k+1,i}$ with weight $w'_{k+1,i}$

compute $x_{k+1} = sum(w'_{k+1,i}\ p'_{k+1,i}) / sum(w'_{k+1,i})$

Figure 2.3. Sequence of Vehicle Tracking Particle Filter

2.3 Process model Input Update with Kalman Filter

In constructing vehicle tracking filter, though the filter tracks the vehicle with zero-input model(eq. $v_k = 0, \dot{\psi} = 0$), if the process model have huge variation between the measurement, the tracking performance gets bad and the total summation of the weight value becomes near zero which occurs computational error while filter is working. From these reasons, estimating inputs and recursively updating those gives better resampling quality and tracking performance. Still, the target vehicles are one of the surrounding environment of the ego-vehicle, which means without any V2V solutions, we cannot access to the actual input values of the target vehicle. Thus, by designing estimator to update the particle filter's inputs for each step. To design estimator, simple Kalman filter is used. But Kalman filter is linear filter, nonlinearity of the system makes large tracking error and fault detection cases. Thus, it is better to estimate the yaw rate input instead of estimating steering angle input directly, otherwise the measurement model of the filter will have inverse function of tangent and inverse of the velocity which is also the estimation target variable.

For estimating the inputs, we design two separate Kalman filter. One is for the velocity estimation and the other one is for yaw rate input estimation. The state variables are defined as $x_{vel,k} = [\ v_k \quad \dot{v}_k\ ]^T$, $x_{steer,k} = [\ \psi_k \quad \dot{\psi}_k \quad \ddot{\psi}_k\ ]^T$ each. For each filter, the measurement information should be generated to update the filter. As for velocity filter, compute the instantaneous velocity from the position information. For the yaw rate filter, yaw angle is directly used for the update. The equation for computing instantaneous velocity from the position estimation results are follows.

Define output of the particle filter

$$
\begin{aligned}
&z_{pos,k} = [\ x_k \quad y_k\ ]^T,\ z_{yaw,k} = \psi_k \\
&z_{vel,k} = norm((z_{pos,k} - z_{pos,k-1}) / \Delta t) \\
&z_{steer,k} = z_{yaw,k}
\end{aligned}
\tag{2.4}
$$

After computing the measurement for velocity & yaw rate filter, the filter is constructed with the following process model. In designing process model, we assumed that the acceleration and the yaw rate as constant. Especially for the yaw rate estimator, estimating with yaw rate and the derivative of yaw rate term gives more noise outputs, which can estimate the yaw rate into wrong direction. Thus, by using yaw angle and yaw rate information, even though the estimation target variable is yaw rate, because the main purpose of the inputs estimation is to track rough scale estimation of inputs, which means the intention of the target vehicle such as left turn or right turn. The process models of two filter are designed as follow.

$$
\begin{aligned}
&x_{vel,k+1} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_{vel,k} \\
&x_{steer,k+1} = \begin{bmatrix} 1 & \Delta t & 0.5\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} x_{steer,k} \\
&x_{vel,k} = [\ v_k \quad \dot{v}_k\ ]^T,\ \ x_{steer,k} = [\ \psi_k \quad \dot{\psi}_k \quad \ddot{\psi}_k\ ]^T \\
&H_{vel} = [\ 1 \ \ 0\ ],\ H_{steer} = [\ 1 \ \ 0 \ \ 0\ ]
\end{aligned}
\tag{2.5}
$$

As mentioned before, the measurement of the filters are instantaneous velocity and yaw angle, thus the measurement relation matrix H_{vel}, H_{steer} are becomes like the upper equation. The actual Kalman filter is constructed with the following process updates & measurement updates equations.

- Process Update Equation

$$\bar{x}_{vel,k+1} = A_{vel}\hat{x}_{vel,k}\ ,\ \bar{x}_{steer,k+1} = A_{steer}\hat{x}_{steer,k}$$
$$M_{vel,k+1} = A_{vel}P_{vel,k}A^{T}{}_{vel} + Q_{vel} \tag{2.6}$$
$$M_{steer,k+1} = A_{steer}P_{steer,k}A^{T}{}_{steer} + Q_{steer}$$

- Measurement Update Equation

$$P^{-1}{}_{vel,k+1} = M^{-1}{}_{vel,k+1} + H^{T}{}_{vel}V^{-1}{}_{vel}H_{vel}$$
$$P^{-1}{}_{steer,k+1} = M^{-1}{}_{steer,k+1} + H^{T}{}_{steer}V^{-1}{}_{steer}H_{steer}$$
$$K_{vel,k+1} = P^{-1}{}_{vel,k+1}H^{T}{}_{vel}V^{-1}{}_{vel} \tag{2.7}$$
$$K_{steer,k+1} = P^{-1}{}_{steer,k+1}H^{T}{}_{steer}V^{-1}{}_{steer}$$
$$\hat{x}_{vel,k+1} = \bar{x}_{vel,k+1} + K_{vel,k+1}\left(z_{vel,k} - H_{vel}\bar{x}_{vel,k+1}\right)$$
$$\hat{x}_{steer,k+1} = \bar{x}_{steer,k+1} + K_{steer,k+1}\left(z_{steer,k} - H_{steer}\bar{x}_{steer,k+1}\right)$$

After estimating the states from the velocity filter and yaw rate filter, extract the input values that the particle filter needs and apply them for each time step updates. The input update is applied with the simple equations as follows.

$$v_{action,k+1} = [\ 1\ \ 0\ ]\hat{x}_{vel,k+1}$$
$$\dot{\psi}_{action,k+1} = [\ 0\ \ 1\ \ 0\ ]\hat{x}_{steer,k+1} \tag{2.8}$$

By following the above sequences, the total vehicle tracking filter is designed to estimate the surrounding vehicles' position and some information to predict their future movement.

Chapter 3

Simulation

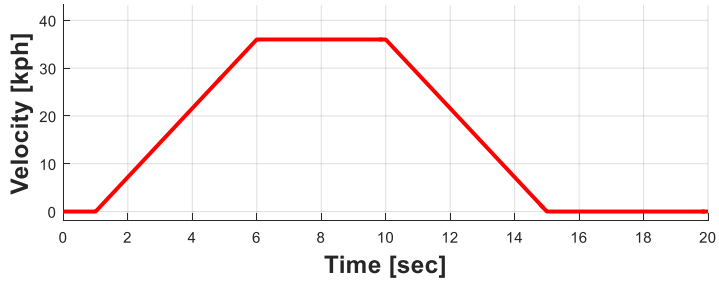
3.1 Pre-defined Profile based Simulation

To verify the algorithm's performance, the simulation is held for three different scenarios. To show the previous surrounding vehicle tracking filter's performance, we simulate with the straight road driving case and left turning vehicle perceive. The first scenario is considered to test the tracking filter can detect the vehicles in basic situations. In this scenarios, as this study's target environment is city condition, we assumed the velocity area near 30kph. But still, at the high speed condition such as high way road, the road usually have smooth curvature, which is not different from the suggested scenario.

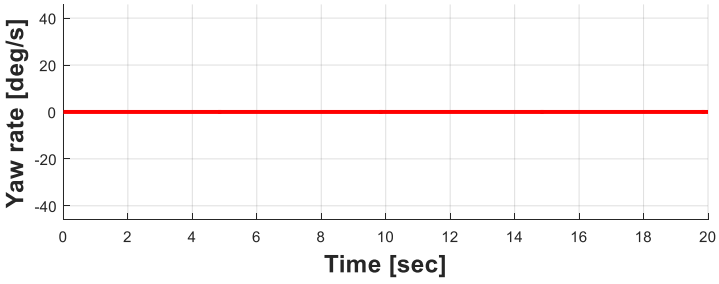
The second scenario is one of the main purpose of this work, which test the performance of the vehicle at the in-city condition such as intersections. Turning motion is the one of the major component of the urban road driving or parking, thus the second scenario is evaluated. Assuming the turning in-city condition, target vehicle motion is considered the situation that slow down and turning with constant steering and after the turning, speed up again.

For each scenarios, as for the reference, the velocity profiles and the yaw rate value are designed as follows. Also, to check the velocity estimation performance, both

scenarios have varying velocity profile. Especially for the straight line case, in urban condition, stop & go shape profile is considered. When generating the simulation, the actual position of the target vehicle is computed from the velocity and yaw rate profile proposed in Figure 3.1-2, and the point cloud data is generated based on the laser scanners' characteristic with noise, and our algorithm is applied to the data.



(a)



(b)

Figure 3.1. Input Profile of Target Vehicle for Case 1

(a) Velocity Profile (b) Yaw rate Profile

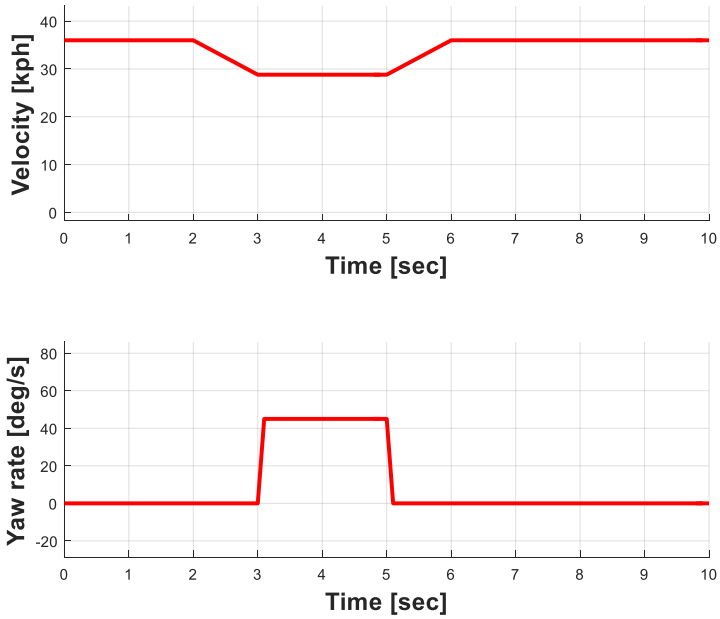


Figure 3.2. Input Profile of Target Vehicle for Case 2

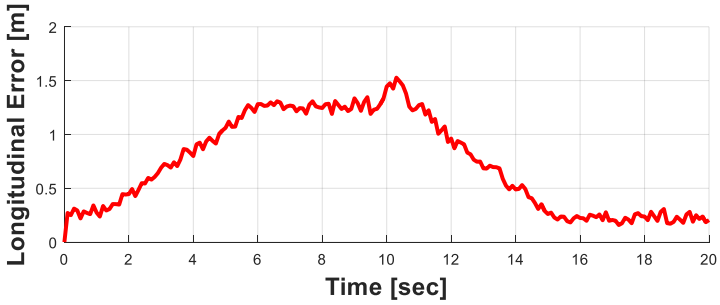
(a) Velocity Profile (b) Yaw rate Profile

The simulation results are proposed in Figure 3.3-6., which show the lateral error, longitudinal error, and the heading angle error each. Both Results show that position estimation error is bounded with near 20cm range, and the heading angle tracking error is bounded with near 15 deg. Especially for the test scenario 2, the left turning case, in 0-1sec area is the filter convergence occurred because of the shape extraction algorithm gives the directly opposite heading angle. But still, during the simulation, the estimation results didn't diverge (eq. fault detection). Even though the simulation yaw rate profile is extreme case which has almost discrete yaw rate change. Thus in both cases, the algorithm tracks the vehicle well.

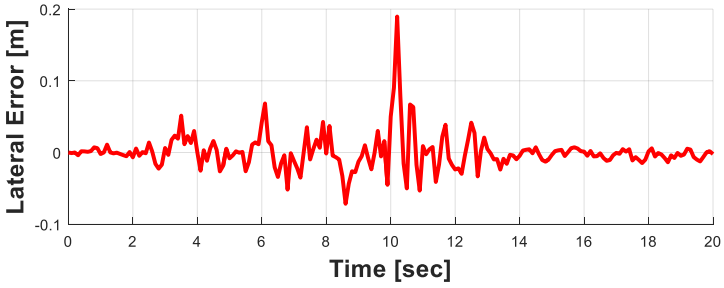
In Figure 3.4, and 3.6., the graphs show the estimation result from the input

estimating Kalman filter. In both cases, the velocity input is well estimated even though the velocity are varying during the simulation. Especially for the test scenario 1, the target vehicle starting from 0 kph and reach to 36 kph and goes down again, the filter shows its performance to track this velocity profile. It tracks the velocity for the scenario 2, except for the region in filter converging time region (0-1sec).

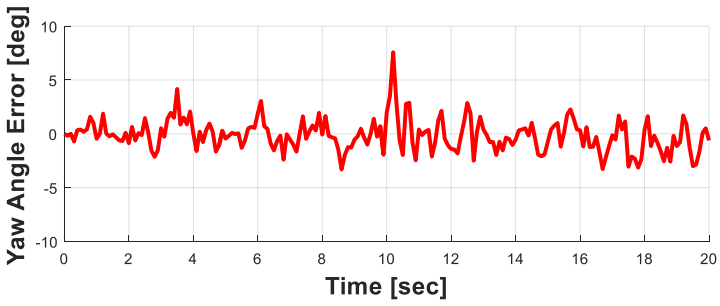
For the yaw rate estimator, Figure 3.4(b) shows that yaw rate estimation works with less noise level and didn't occur any fault detection cases. To see the tracking performance, shown in Figure 3.6(b), the estimator didn't track the exact yaw rate, however, the estimator shows first order delayed motion of the actual value and gives the right direction of the yaw rate change which can be used in both target vehicle intention prediction information and process model update for particle filter.



(a)



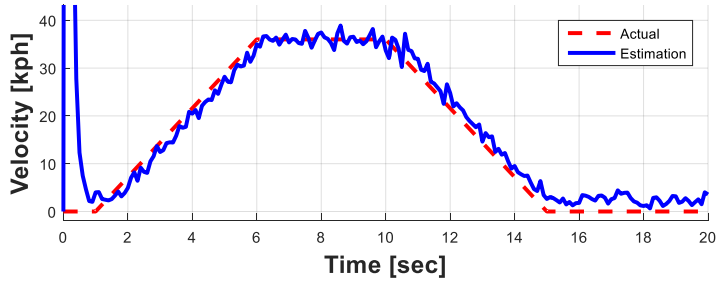
(b)



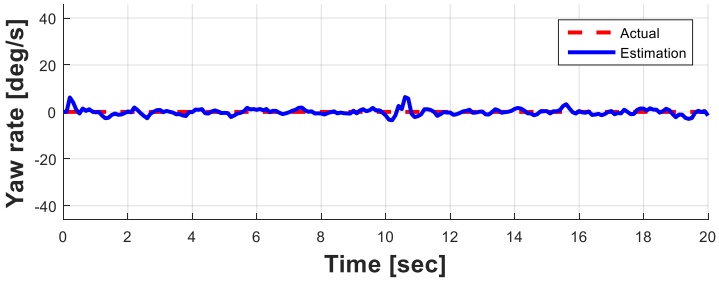
(c)

Figure 3.3. Position Estimation of Target Vehicle for Case 1

(a) Longitudinal error (b) Lateral error (c) Heading angle error



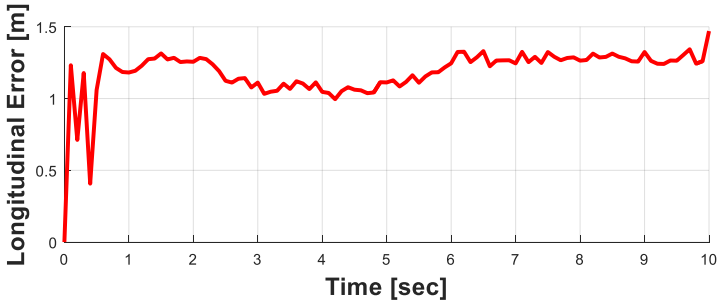
(a)



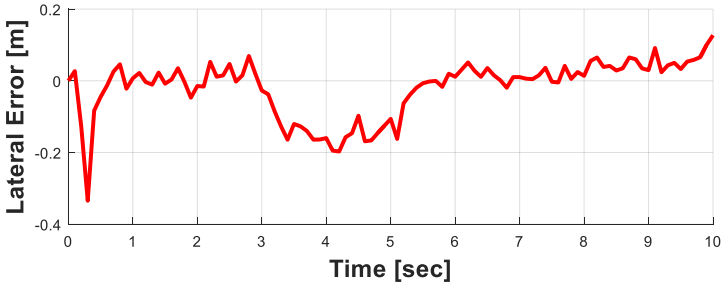
(b)

Figure 3.4. Input Estimation of Target Vehicle for Case 1

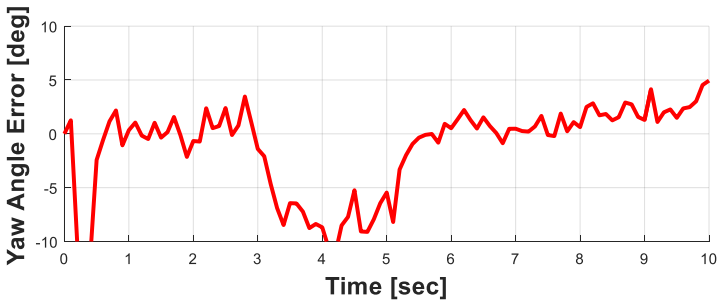
(a) Velocity Estimation (b) Yaw rate Estimation



(a)



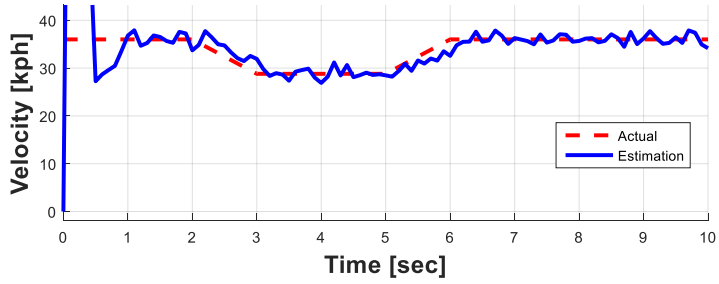
(b)



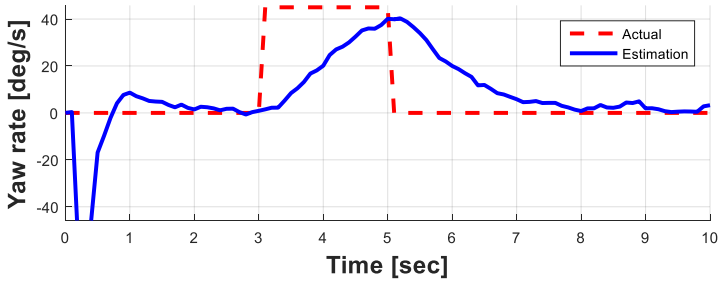
(c)

Figure 3.5. Position Estimation of Target Vehicle for Case 2

(a) Longitudinal error (b) Lateral error (c) Heading angle error



(a)



(b)

Figure 3.6. Input Estimation of Target Vehicle for Case 2

(a) Velocity Estimation (b) Yaw rate Estimation

3.2 Intersection Environment based Simulation

For the third scenario, as a main simulation for this study, simulation with two vehicle in intersection is held. The simulation condition of the surroundings are shown in Figure 3.7. The ego-vehicle is set on the left side of the intersection, and the two target vehicles are coming from north and south direction each. Both target vehicles turn left at the intersection.

The result of the intersection simulation is shown in Figure 3.8-11 (vehicle 1 : Figure 3.8-9., vehicle 2 : Figure 3.10-11.). For both result, the total position error is bounded with maximum 50cm distance error. Especially for the lateral error, which is one of the most important state of designing automated driving controller, has bounded with almost 30cm level. This error level doesn't harm the accuracy of the lane information of the vehicle, thus the proposed algorithm can be evaluated with the automated vehicle. At the same time, for the heading angle of the target vehicle, the error is bounded under 10 degree, which is also acceptable amount of error in lane keeping driving sequence. From Figure 3.9. and 3.11., the velocity estimator almost tracks the actual values, and yaw rate estimator tracks the actual value with a bit of fluctuation. Still, even for yaw rate estimation result, the error between the actual value and the estimated value doesn't have enough amount to yield fault decision for the tendency of input values' variation. Thus the estimated input variables also have potential to become useful information to decide the surrounding driver's intention additionally to the position information that are estimated from vehicle tracking filter which is proposed.

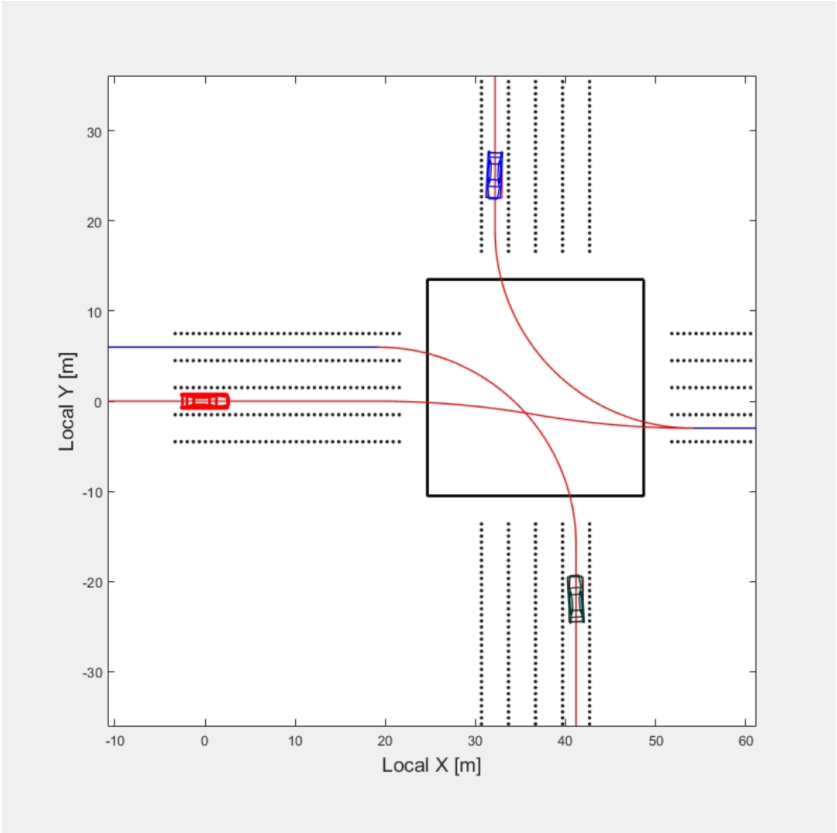
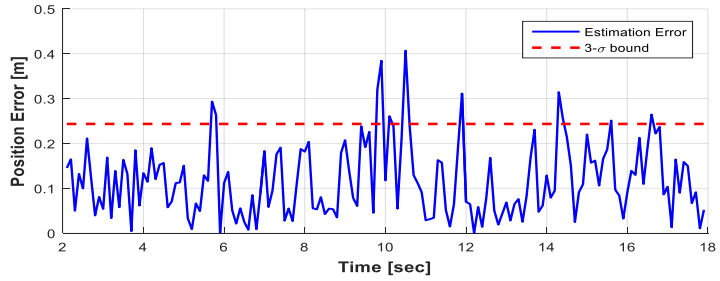
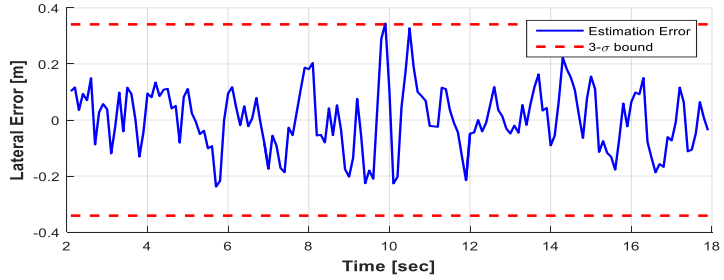


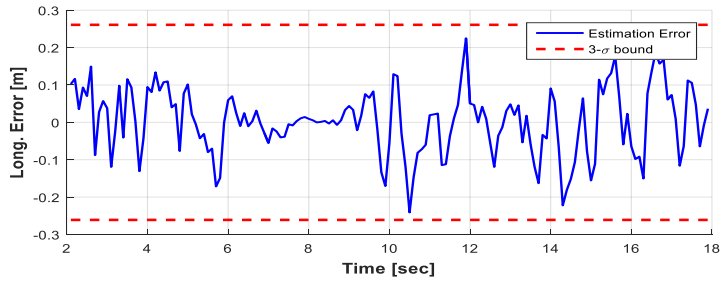
Figure 3.7. Condition for Intersection Simulation



(a)



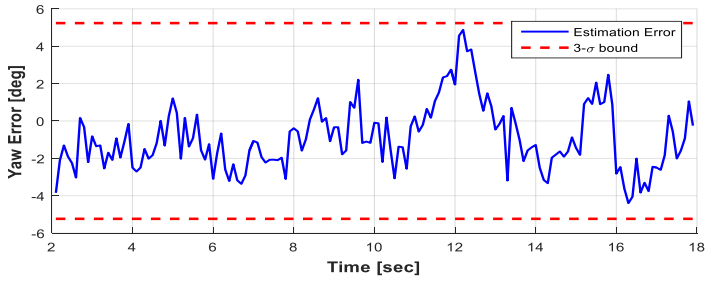
(b)



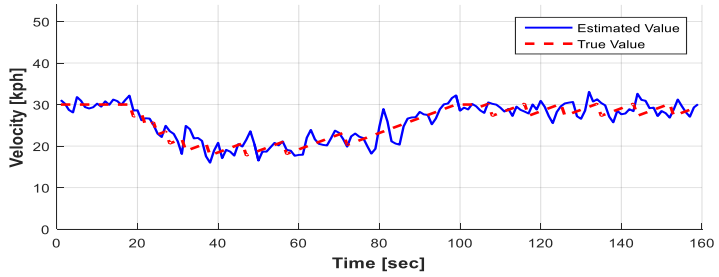
(c)

Figure 3.8. Tracking Error of Target Vehicle 1

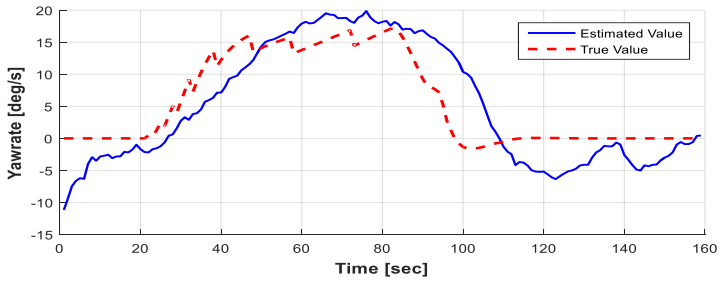
(a) Position Error (b) Lateral Error (c) Longitudinal Error



(a)



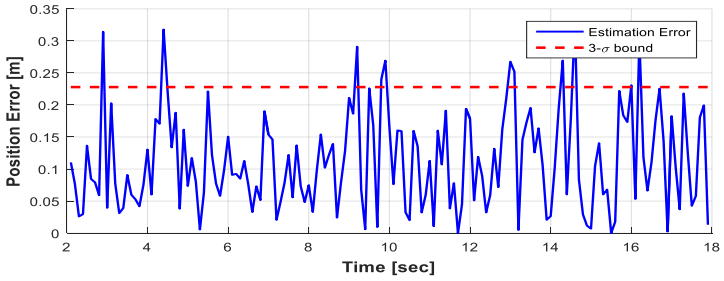
(b)



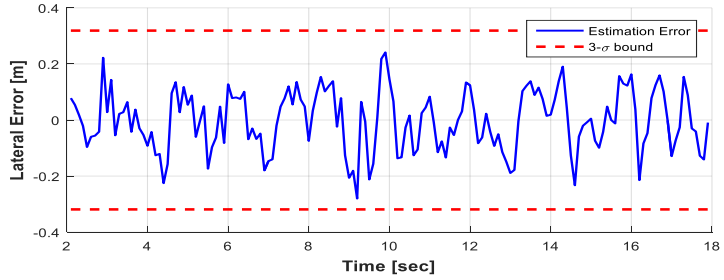
(c)

Figure 3.9. Yaw Error & Input Estimation of Target Vehicle 1

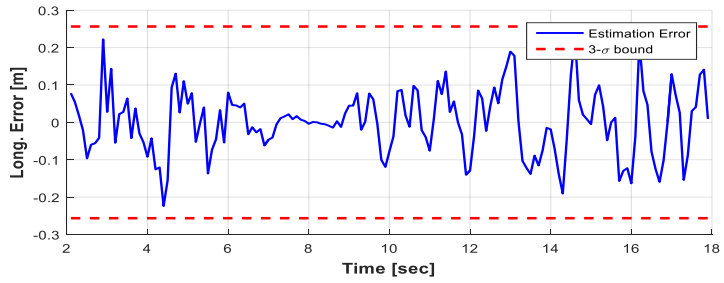
(a) Yaw Error (b) Velocity Estimation (c) Yaw rate Estimation



(a)



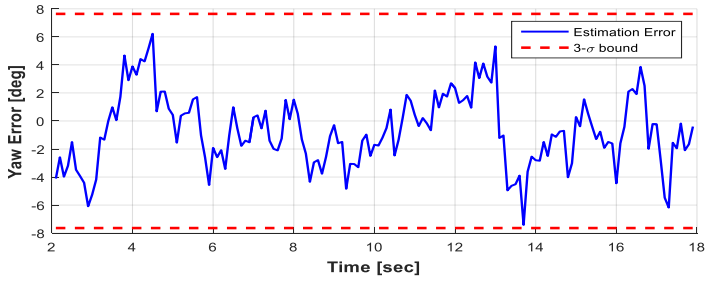
(b)



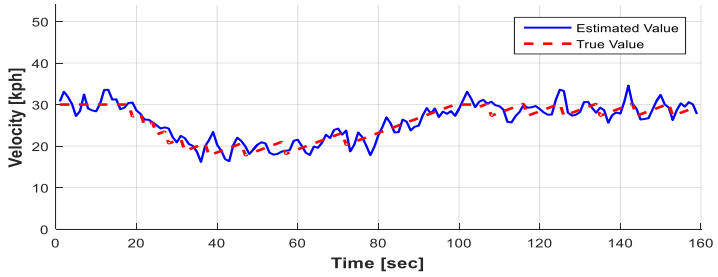
(c)

Figure 3.10. Tracking Error of Target Vehicle 2

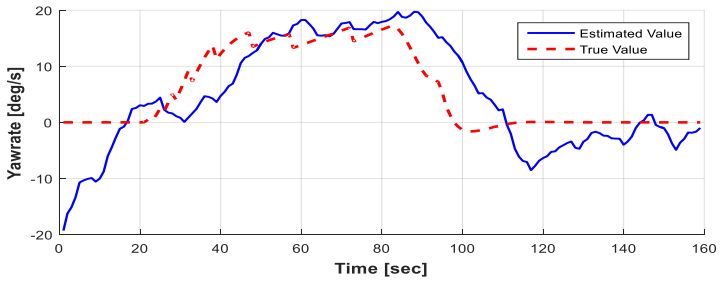
(a) Position Error (b) Lateral Error (c) Longitudinal Error



(a)



(b)



(c)

Figure 3.11. Yaw Error & Input Estimation of Target Vehicle 2

(a) Yaw Error (b) Velocity Estimation (c) Yaw rate Estimation

Chapter 4

Vehicle Experiment Data Test

4.1 Intersection Environment based Simulation

For verifying the performance of the proposed algorithm, as a next step of the simulation, we evaluated simulation based on vehicle experiment data and the actual vehicle test. As a first stage, for the vehicle experiment data simulation, we used Hyundai Avante (Elantra) with four laser scanner (Sick LMS511), two for localization information and two for the surrounding obstacle detection. The sensor configuration and field of view of the test vehicle is shown in figure 4.1.

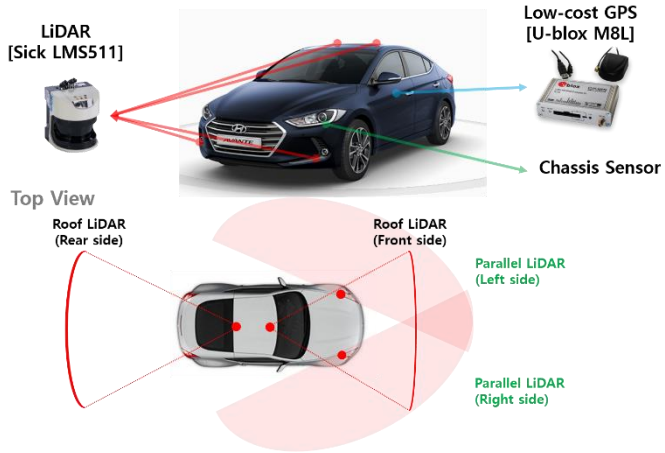


Figure. 4.1. Sensor Configuration & FOV of Avante

For the actual vehicle test, Hyundai IONIQ is used. The vehicle has six laser scanner, the IBEO Lux for detecting surroundings and collecting reference information efficiently in urban scenarios. Configuration and field of view of the vehicle is shown in figure 4.2. also.

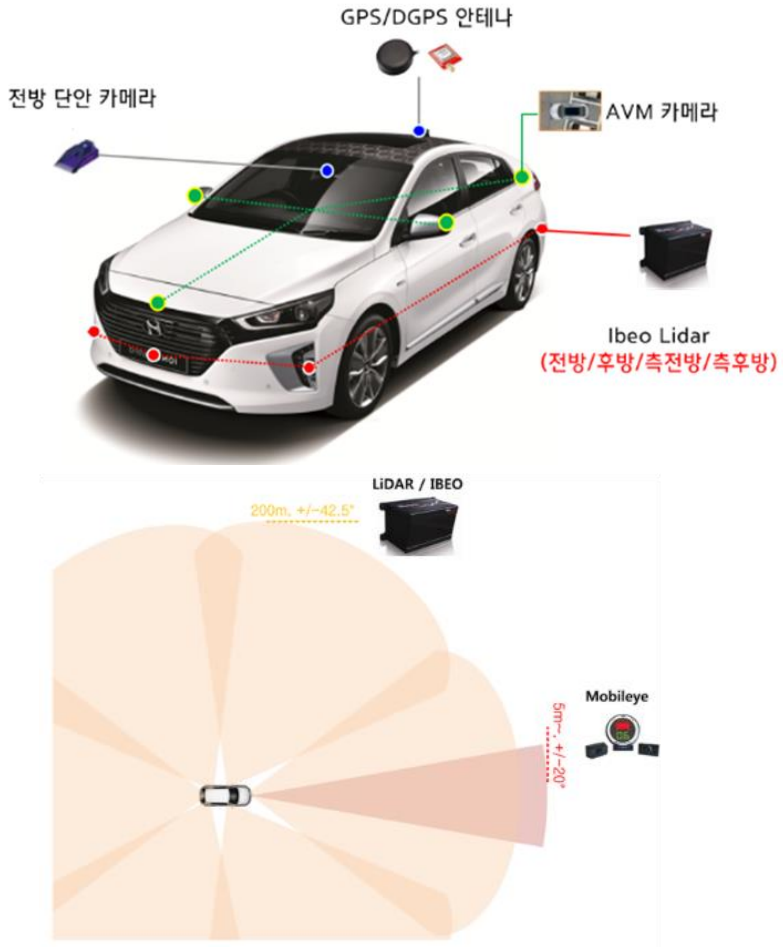


Figure. 4.2. Sensor Configuration & FOV of IONIQ

4.2 Intersection Environment based Simulation

For the testament data based simulation, we collected the laser scanner point data for two different scenarios. One is for the varying velocity condition, and the other is for heading angle varying scenario (e.q.Turning). For the first scenario, we used preceding vehicle which is driving with constant velocity, and suddenly stop for a while, and go straight again. The second scenario is waiting for parking out vehicle until the target vehicle comes out and finishing their heading aligning and go. These scenarios are shown in figure 4.3.

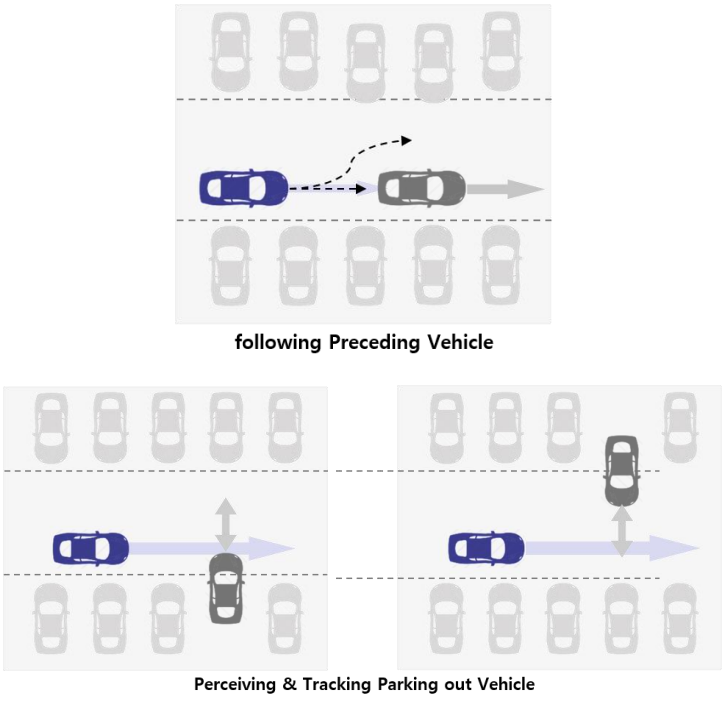
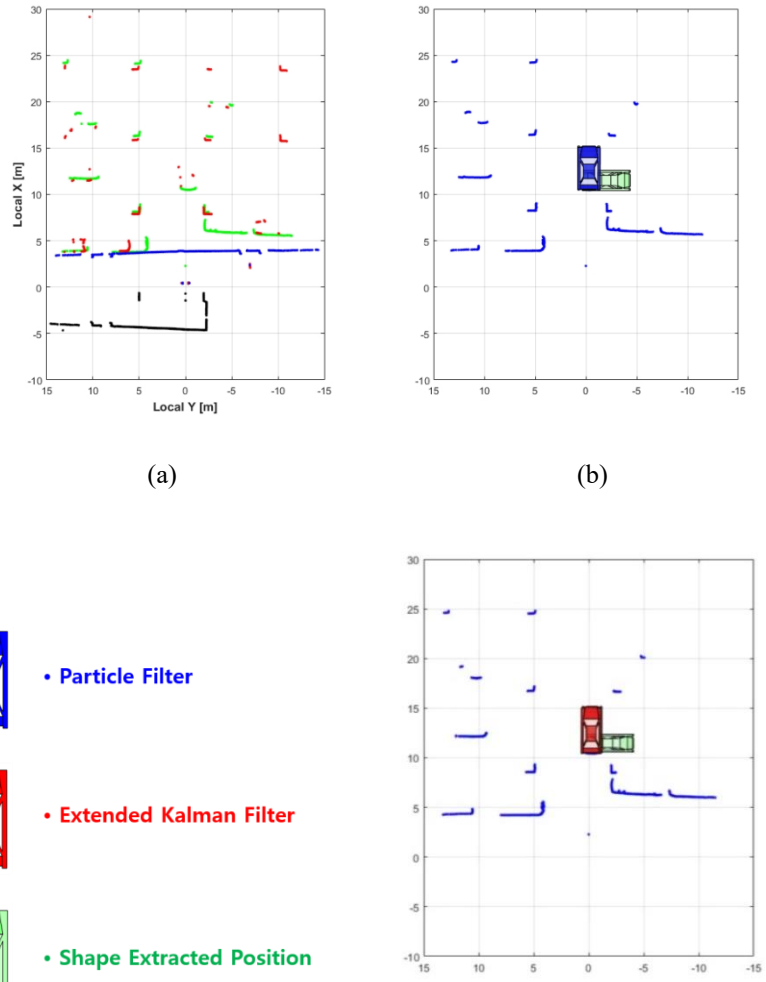


Figure. 4.3. Test Scenarios for Experimental data based Simulation

The test result is shown in figure 4.4-11. with snapshots of the test data with raw laser scanner point cloud data and tracking result of particle filter and model based Extended Kalman filter as a comparison. For each snapshot, the left-top figure (a) shows the raw data, and the right-top figure (b) shows the tracking result with particle filter and the right-bottom figure (c) shows the result with EKF. For figure (b) and (c), shape extraction results also shown as a brighter vehicle. Figure 4.4-7. show the result for the first scenario, and the results show that EKF occurring tracking delay when the target's motion have variation from the assumed model. At the same time, particle filter adapting the varying motion faster. The difference of their convergence can be exactly shown in figure 4.6.

As for the second scenario, the test results are shown in figure 4.8-11. Comparing the two different filter, the result of the EKF shows that it has troubles in their filter converging time and filter divergence issue when they updated with unexpected motion information. However, for the particle filter, the filter tracks the turning motion well and tracks faster than the EKF. Thus, results of both scenario shows that proposed algorithm (particle filter with input update) can track better than EKF which is commonly used for tracking filter.



(a) (b)

(c)

Figure. 4.4. Snapshot for Scenario 1 (t = 0s)

(a) Raw data (b) Particle filter based tracking (c) EKF based tracking

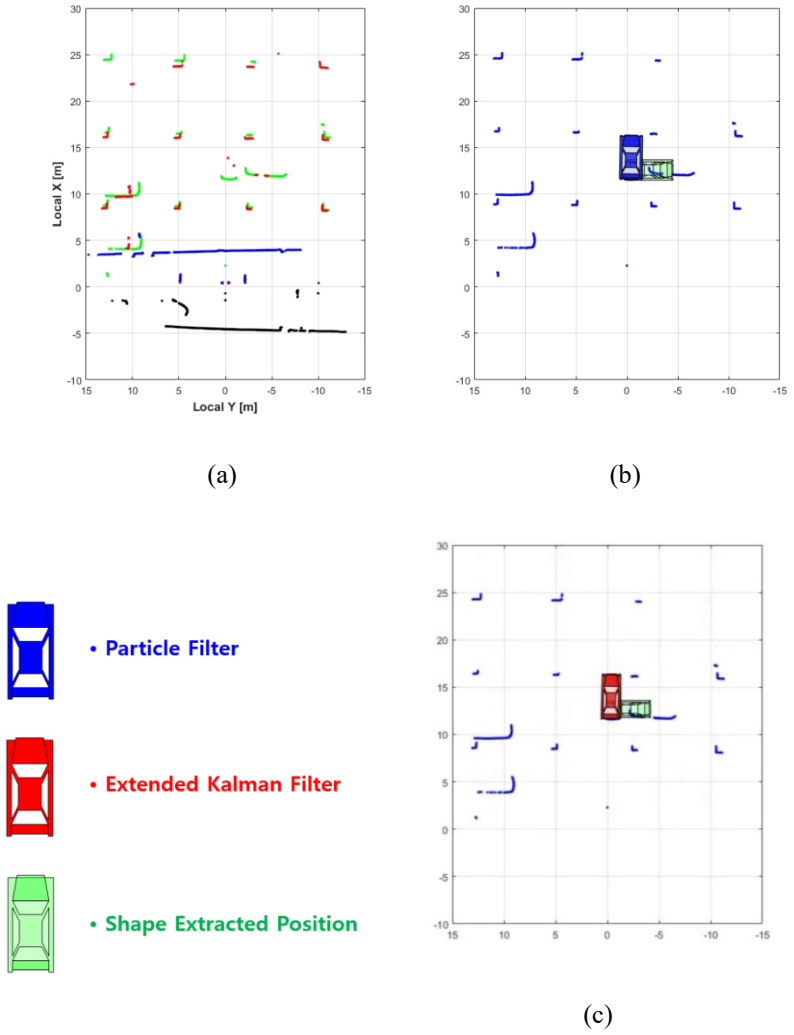
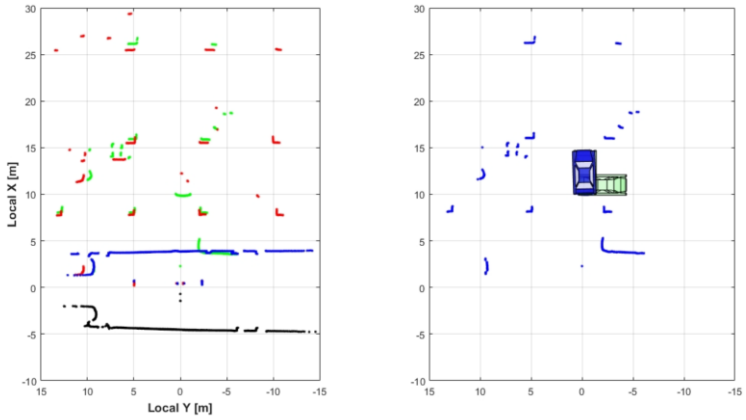
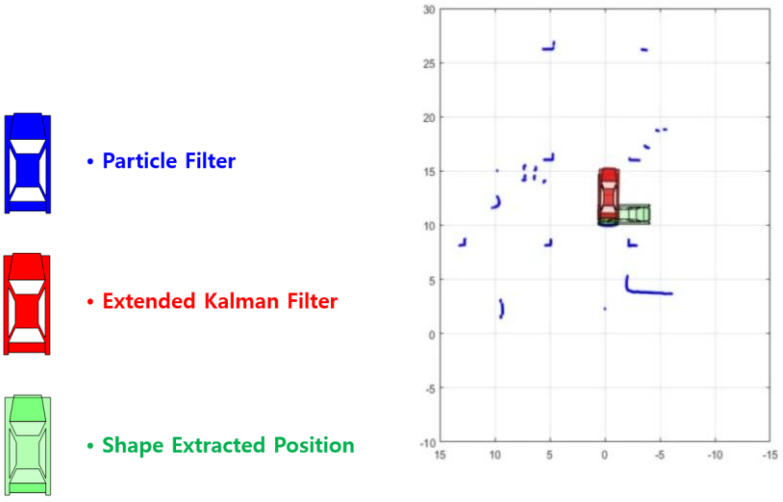


Figure. 4.5. Snapshot for Scenario 1 (t = 5s)

(a) Raw data (b) Particle filter based tracking (c) EKF based tracking



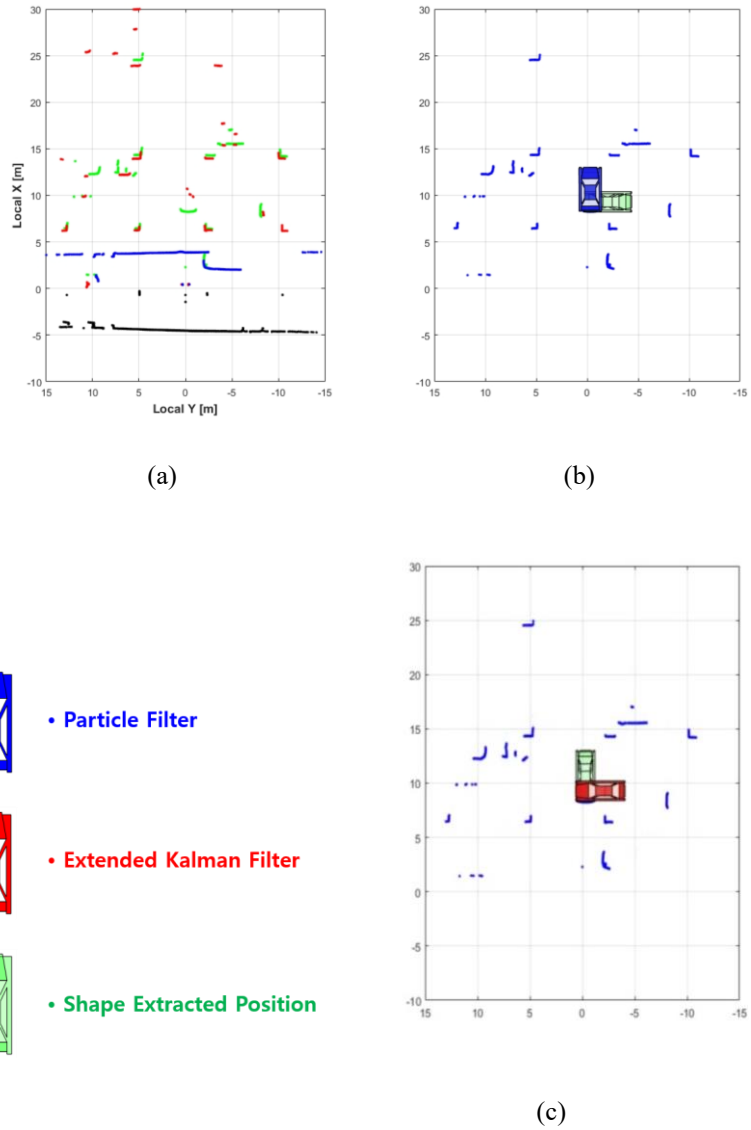
(a) (b)



(c)

Figure. 4.6. Snapshot for Scenario 1 (t = 10s)

(a) Raw data (b) Particle filter based tracking (c) EKF based tracking

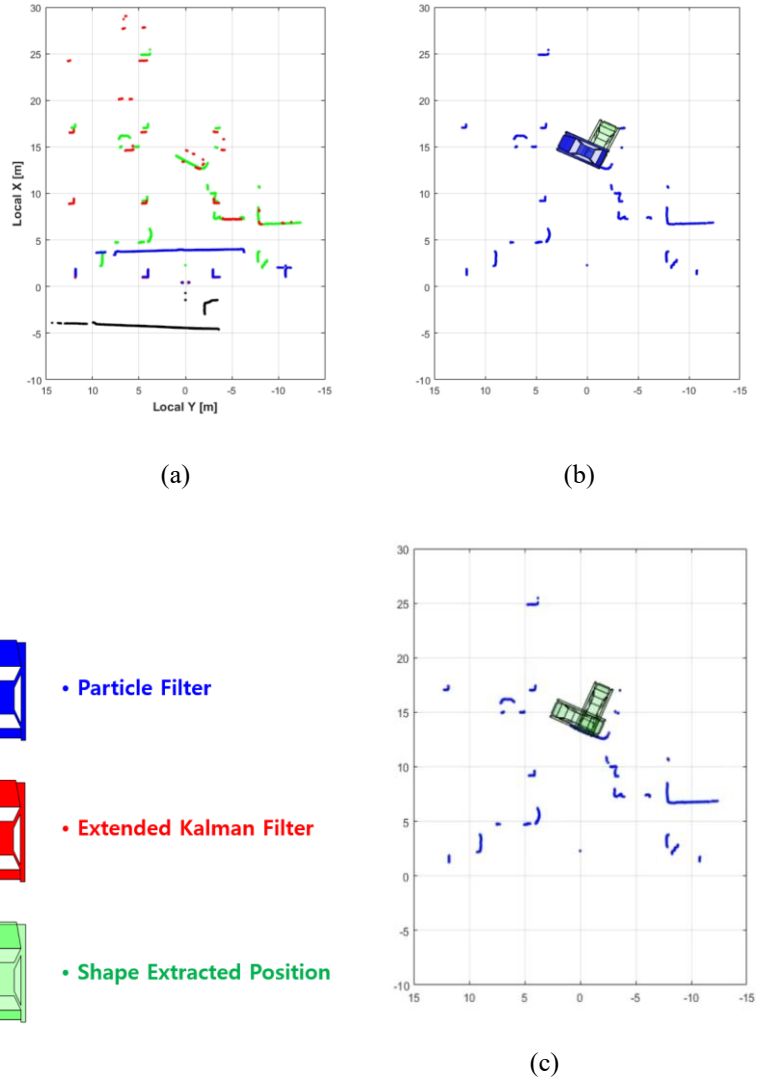


(a) (b)

(c)

Figure. 4.7. Snapshot for Scenario 1 (t = 15s)

(a) Raw data (b) Particle filter based tracking (c) EKF based tracking



(a) (b)

(c)

Figure. 4.8. Snapshot for Scenario 2 (t = 0s)

(a) Raw data (b) Particle filter based tracking (c) EKF based tracking

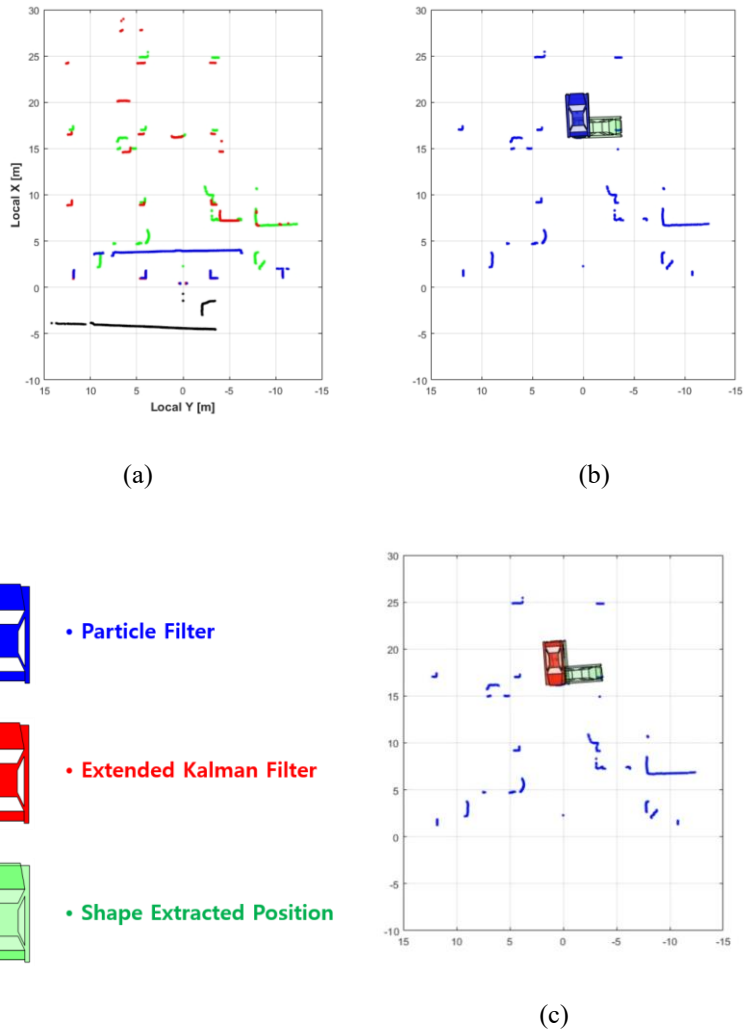


Figure. 4.9. Snapshot for Scenario 2 (t = 3s)

(a) Raw data (b) Particle filter based tracking (c) EKF based tracking

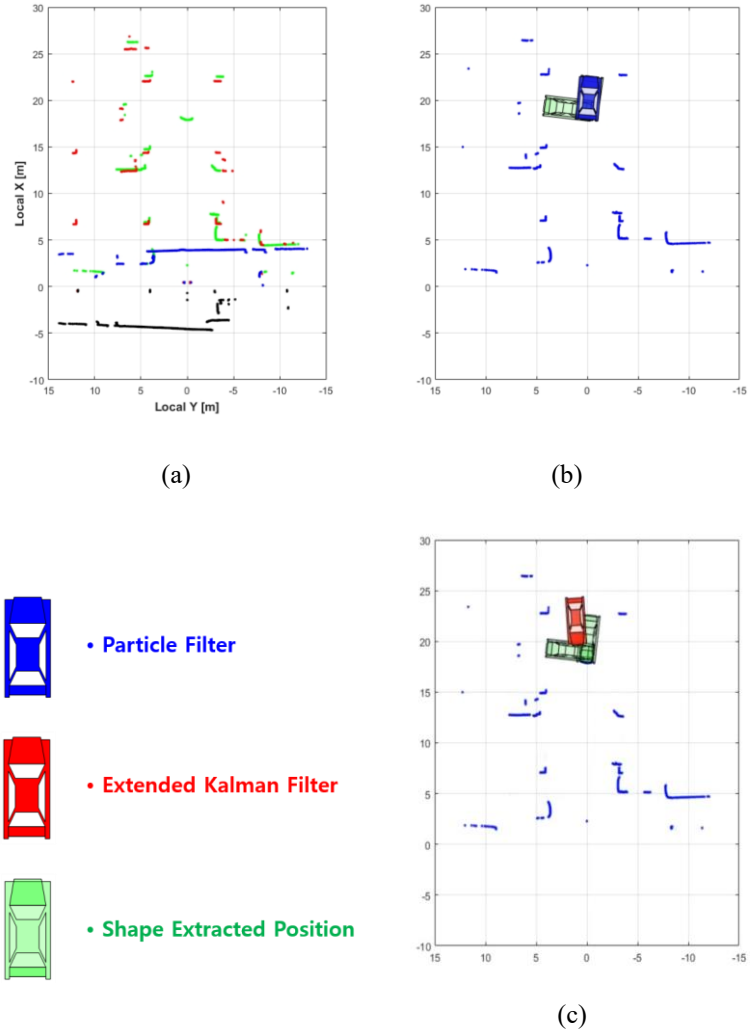
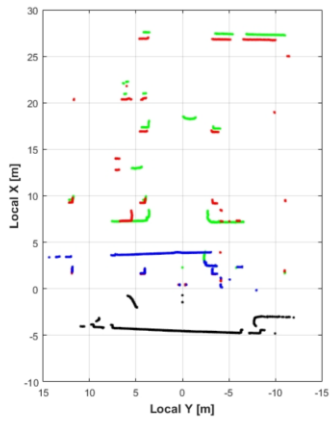
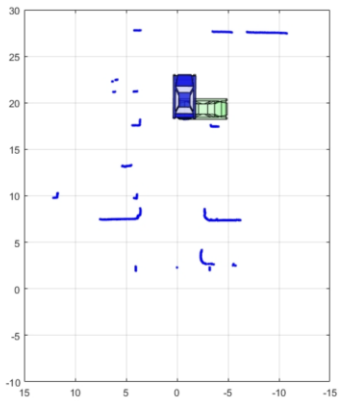


Figure. 4.10. Snapshot for Scenario 2 (t = 6s)

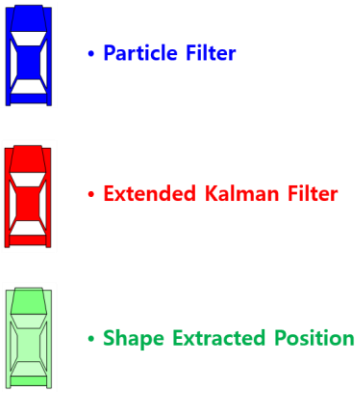
(a) Raw data (b) Particle filter based tracking (c) EKF based tracking

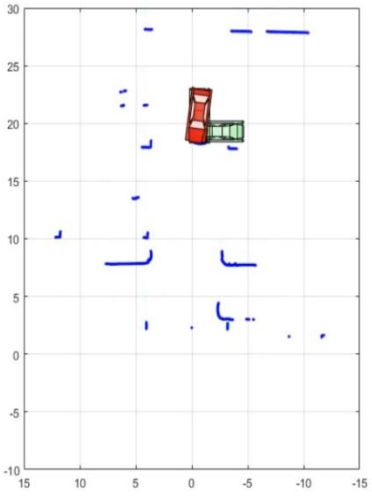


(a)



(b)





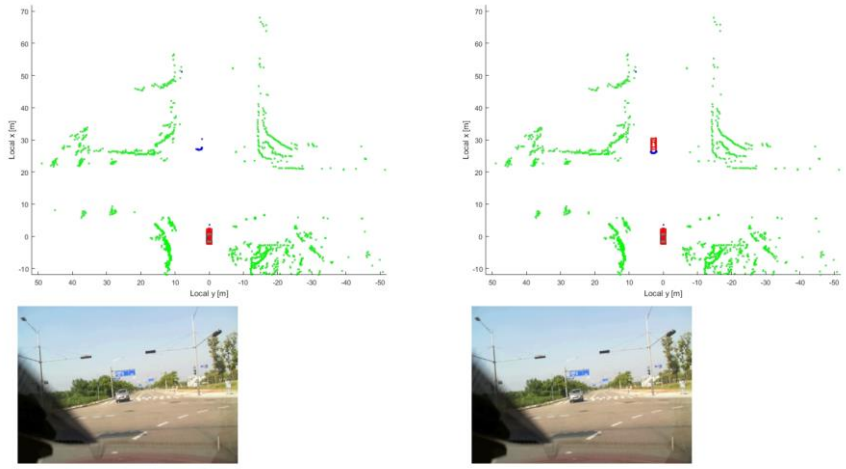
(c)

Figure. 4.11. Snapshot for Scenario 2 (t = 9s)

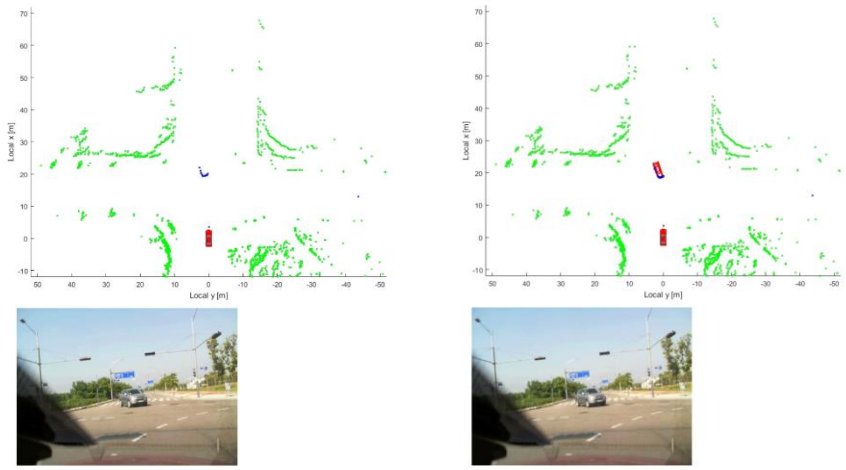
(a) Raw data (b) Particle filter based tracking (c) EKF based tracking

4.3 Intersection Environment based Simulation

As mentioned in the section 4.1., we tested the performance of the proposed algorithm with Hyundai IONIQ which is shown in figure 4.2. For the test, the ego-vehicle stop at the intersection and waiting for surroundings coming to the intersection. The algorithm was proposed with industrial system level personal computer with window OS and commercialized software Matlab & Labview. The Tracking results shows for two different target vehicle, including going-straight vehicle and turning vehicle. The first case is detecting the left turning vehicle coming from in front of the ego-vehicle. The second case is detecting going-straight vehicle which is coming from the left side of the intersection and go through it. Both scenarios are selected that can frequently occur at the intersection condition. The actual test is held in intersection which is located at MIDAN city in Incheon. As same as section 4.2, the results are proposed as snapshots with raw data of the condition and result of the suggested algorithm. From the results of both test scenarios, the filter tracks well for both varying their heading angle cases and moving in right angle to the ego-vehicle, except for the initial moment for filter converging.



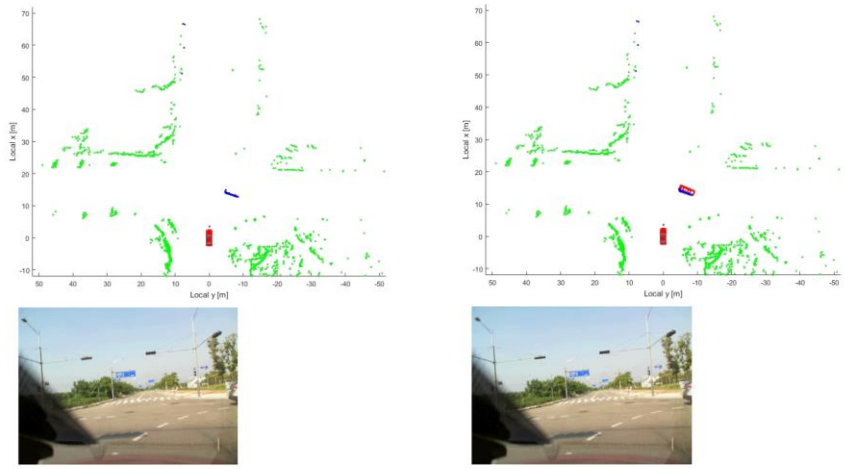
(a)



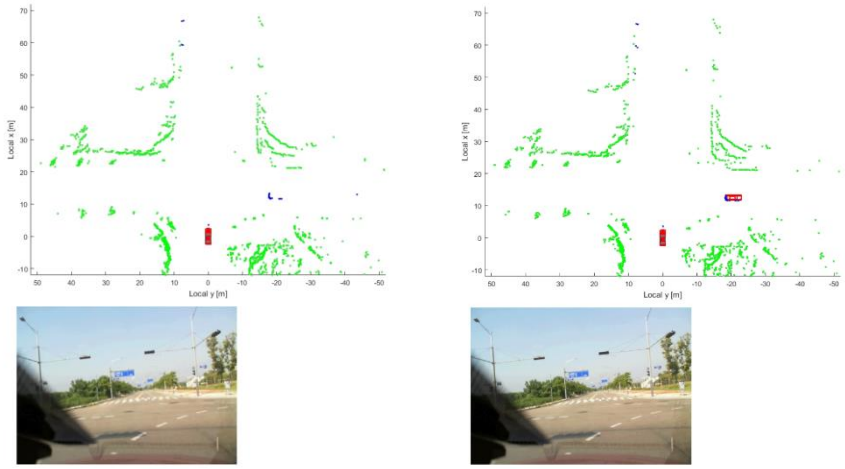
(b)

Figure. 4.12. Snapshot for Test 1 at 0s & 2s

(a) T = 0s moment (b) T = 2s moment (left : raw data, right : particle filter)



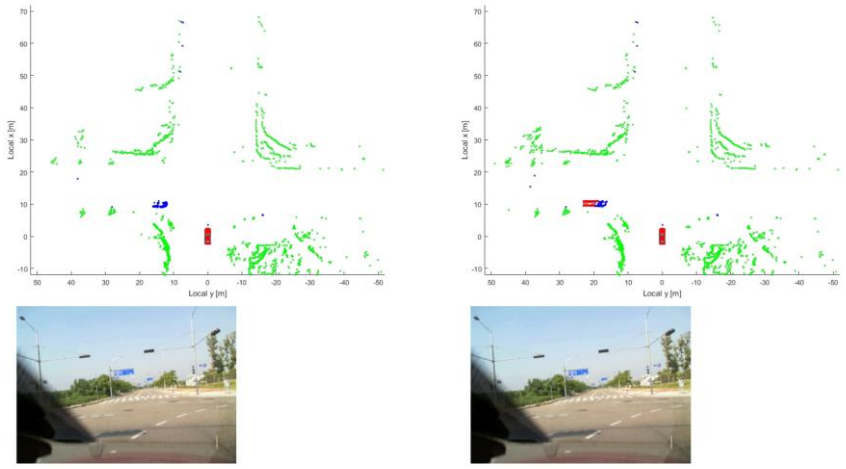
(a)



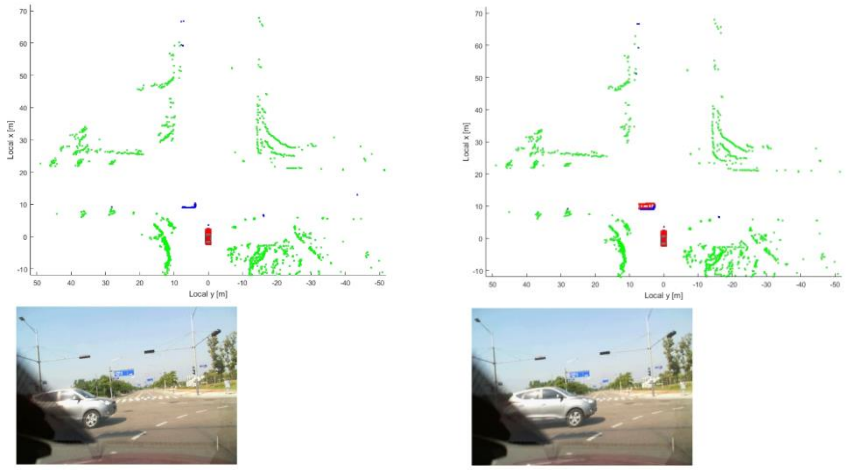
(b)

Figure. 4.13. Snapshot for Test 1 at 4s & 6s

(a) T = 4s moment (b) T = 6s moment (left : raw data, right : particle filter)



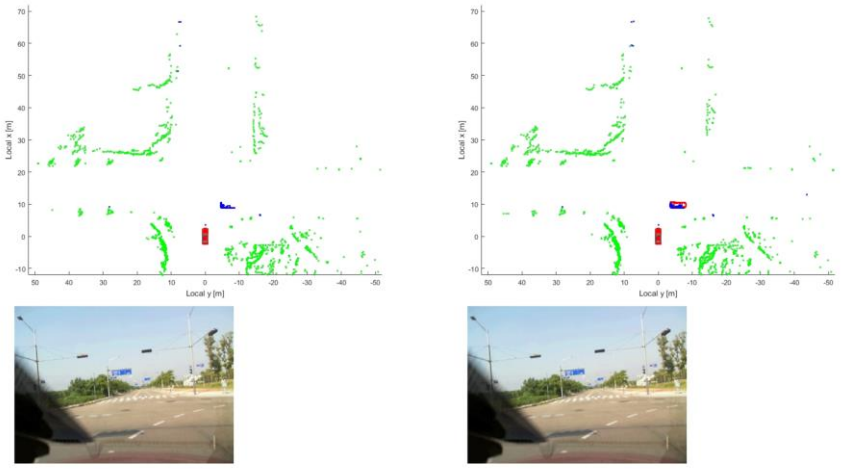
(a)



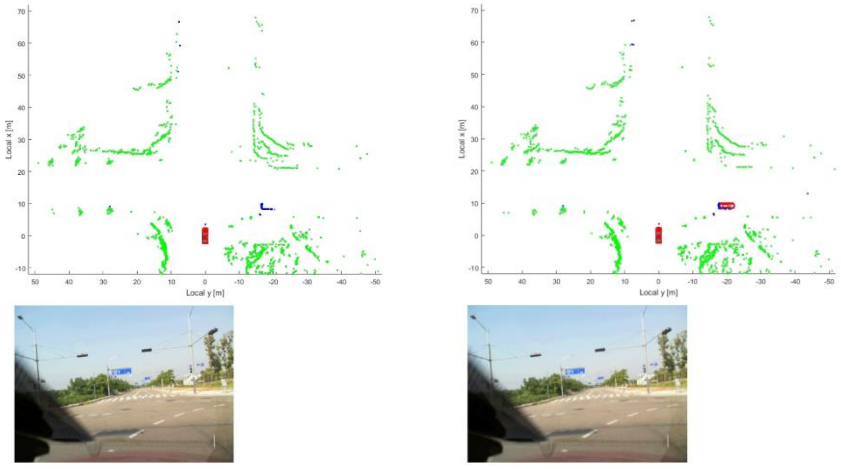
(b)

Figure. 4.14. Snapshot for Test 2 at 0s & 1s

(a) T = 0s moment (b) T = 1s moment (left : raw data, right : particle filter)



(a)



(b)

Figure. 4.15. Snapshot for Test 2 at 2s & 3s

(a) T = 2s moment (b) T = 3s moment (left : raw data, right : particle filter)

Chapter 5

Conclusion

Surrounding vehicle position and velocity variable estimation based on particle filter has been developed. As the filter works with the state position & heading angle information, clustering & shape extraction algorithm has been applied as post-processing algorithm to generate filter measurement from the sensor data. To avoid the tracking filter's process model drift apart from the actual motion far, Kalman filter is recursively designed to update the two inputs, velocity and yaw rate for the process model of the particle filter. The performance of the proposed vehicle tracking filter's tracking ability and input estimating ability are verified via full off-line simulation and vehicle data based simulation, especially for the lateral tracking performance and velocity estimation. In addition, the proposed algorithm evaluated with the actual test vehicle, testing for intersection condition in the urban road. As for the future work, improving the tracking performance and verifying the performance using two test vehicles with high resolution DGPS solutions for both and calculate the error index for the actual cases. At the same time, evaluating with vehicle level at multi-vehicle condition will be considered.

Bibliography

[Enache10] N. M. Enache, S. Mammar, M. Netto, and B. Lusetti, "Driver steering assistance for lane-departure avoidance based on hybrid automata and composite Lyapunov function," Intelligent Transportation Systems, IEEE Transactions on, vol. 11, pp. 28-39, 2010.

[Hoedemaeker98] M. Hoedemaeker and K. A. Brookhuis, "Behavioural adaptation to driving with an adaptive cruise control (ACC)," Transportation Research Part F: Traffic Psychology and Behaviour, vol. 1, pp. 95-106, 1998.

[O'Malley10] R. O'Malley, E. Jones, and M. Glavin, "Rear-lamp vehicle detection and tracking in low-exposure color video for night conditions," Intelligent Transportation Systems, IEEE Transactions on, vol. 11, pp. 453-462, 2010.

[Tingvall00] C. Tingvall and N. Haworth, "Vision Zero: an ethical approach to safety and mobility," in 6th ITE International Conference Road Safety & Traffic Enforcement: Beyond, 2000.

[Schulze05] M. Schulze, G. Nocker, and K. Bohm, "PReVENT: A European program to improve active safety," in Proc. of 5th International Conference on Intelligent Transportation Systems Telecommunications, France, 2005.

[Wang12] Wang D.Z., Posner I., Newman P., "What could move? Finding cars, pedestrians and bicyclists in 3D laser data." Robotics and Automation (ICRA), 2012 IEEE International Conference on, doi:10.1109/ICRA.2012.6224734

[Jeong16] Jeong, Yonghwan, et al. Design and Implementation of Parking

Control Algorithm for Autonomous Valet Parking. No. 2016-01-0146. SAE Technical Paper, 2016.

[Derpanis10] Derpanis, Konstantinos G. "Overview of the RANSAC Algorithm." Image Rochester NY 4.1 (2010): 2-3

[Kaempchen04] Kaempchen, Nico, et al. "Sensor fusion for multiple automotive active safety and comfort applications." Advanced Microsystems for Automotive Applications 2004. Springer Berlin Heidelberg, 2004. 137-163.

[Won10] Won, Seong-hoon Peter, Wael William Melek, and Farid Golnaraghi. "A Kalman/particle filter-based position and orientation estimation method using a position sensor/inertial measurement unit hybrid system." IEEE Transactions on Industrial Electronics 57.5 (2010): 1787-1798.

[Zhang07] Zhang, Bai, et al. "Particle-filter-based estimation and prediction of chaotic states." Chaos, Solitons & Fractals 32.4 (2007): 1491-1498.

[Sameh13] Sameh Mejri, Ali Sghaier Tlili and Naceur Benhadj Braiek, “Particle Filter for State and Unknwon Input Estimation of Chaotic Systems”, International Conference on Control, Engineering & Information Technology (CEIT'13), Proceedings Engineering & Technology - Vol.4, pp. 67-72, 2013

[Bolandhemmat12] Bolandhemmat, Hamidreza, Christopher Clark, and Farid Golnaraghi. "A solution to the state estimation problem of systems with unknown inputs." Recent Patents on Mechanical Engineering 5.2 (2012): 102-112.

[Lee08] Lee, Anthony. Towards smooth particle filters for likelihood estimation with multivariate latent variables. Diss. University of British Columbia, 2008.

[Hasan16] Hasan, Yumnah, et al. "Comparative analysis of vehicle detection in urban traffic environment using Haar cascaded classifiers and blob statistics." Future Technologies Conference (FTC). IEEE, 2016.

[Wei17] Wei, Qifan. "Adaptable Vehicle Detection and Speed Estimation for Changeable Urban Traffic with Anisotropic Magnetoresistive Sensor." IEEE Sensors Journal (2017).

[Zhao16] Zhao, Minhui, Chihang Zhao, and Xingzhi Qi. "Comparative analysis of several vehicle detection methods in urban traffic scenes." Sensing Technology (ICST), 2016 10th International Conference on. IEEE, 2016.

[Furgale12] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Muhlfellner, et al., "Toward automated driving in cities using close-to-market sensors: An overview of the V-Charge Project," in Intelligent Vehicles Symposium (IV), 2013 IEEE, 2013, pp. 809-816.

[Rajamani05] R. Rajamani, Vehicle Dynamics and Control, New York, Springer-Verlag, 2005.

[Kim15] B. Kim, K. Yi, H.-J. Yoo, H.-J. Chong, and B. Ko, "An IMM/EKF Approach for Enhanced Multitarget State Estimation for Application to Integrated Risk Management System," Vehicular Technology, IEEE Transactions on, vol. 64, pp. 876-889, 2015.

[Kim14] B. Kim and K. Yi, "Probabilistic and Holistic Prediction of Vehicle States Using Sensor Fusion for Application to Integrated Vehicle Safety Systems," Intelligent Transportation Systems, IEEE Transactions on, vol. 15, pp. 2178-2190, 2014.

초 록

레이저 스캐너 기반 자율주행용 교차로 내 주변 차량 인지 알고리즘 개발

본 연구는 도심 도로에서 주변 차량을 효과적으로 인지하기 위한 알고리즘을 개발하고자 진행된 연구이다. 자율주행 산업계에서는 최근 자율주행 제어기 설계, 차량용 측위 알고리즘 개발, 환경 인지 알고리즘 개발의 크게 3개로 분류하여 활발히 연구가 진행되고 있다. 이중에서 환경 인지는 인지하고자 하는 대상 환경에 따라 센서나 인지 전략을 맞춰 설계하게 되는데, 도심도로에서는 기존에 상용 시스템 등에서 많이 사용되던 레이더 센서나 카메라 센서 등으로 정확하게 감지하지 못하거나, 혹은 감지 자체가 아에 불가능한 경우 등이 생기기도 하며, 주변 장애물들의 위치 정보 정밀도의 중요성이 고속도로나 자동차 전용 도로 에 비해 높아지기 때문에, 레이저 스캐너가 융합하여 시스템을 설계하고 있다. 따라서 이종 센서 간의 융합이나 도심 내 장애물 인지 정확도 개선을 위해 레이저 스캐너를 통한 장애물 추적에 관한 연구가 필요하다.

따라서 본 연구에서는 레이저 스캐너 기반의 주변 차량의 거동 추정 알고리즘을 개발하였다. 본 알고리즘은 크게 2가지 단계로 구성되어 있으며, 각각 레이저 스캐너 데이터의 선처리 작업과 차량 추적 필터이다.

레이저 스캐너의 경우, 동일한 장애물에서도 다수의 점데이터군이 발생하기 때문에, 이와 같은 점데이터군에서 각 장애물을 대표할 대표점을

구분할 필요가 있다. 따라서 점데이터군을 군집화하고, 각 그룹에 대해서 차량의 경우, 형상을 반영하여 대표점 후보군을 생성한다. 이와 같이 생성된 후보군을 측정값으로 차량 추적 필터를 파티클 필터를 사용하여 설계하였으며, 이를 통해 차량의 위치를 추정하였다. 도심도로, 특히 교차로에서는 주변 차량의 거동을 모델 기반으로 예측하는 것은 정확성이 떨어지므로, 파티클 필터에서 추정된 위치 변화를 기반하여 대상 차량의 속도와 요레이트에 대한 칼만 필터를 설계하여 다시 파티클 필터의 입력으로서 반영할 수 있도록 재귀 구조로 설계하였다.

이와 같이 설계된 알고리즘은 시뮬레이션과 실차 데이터 검증, 실차 적용 검증의 3단계를 통해 검증되었다. 먼저 대상 차량의 위치 정보를 계산 가능하기에 정량적인 오차를 제시할 수 있는 시뮬레이션 환경에서는 대상 차량의 입력 프로파일을 지정하여 알고리즘의 추종 성능을 확인하였으며, 본 논문의 대상인 교차로 환경의 시뮬레이터를 설계하고, 대상 차량들에 대해 경로 추종 알고리즘을 간략하게 설계하여 해당 환경 내에서 인지 알고리즘의 추종 성능을 검증하였다. 추가로 레이저 스캐너 장착된 차량을 통해 일부 도심 환경 조건을 취득하고, 해당 데이터 상에서 정상적으로 인지가 동작하는 지에 관한 여부를 실차 데이터 기반 시뮬레이션으로 검증하였으며, 최종적으로 알고리즘을 산업용 PC 상에서 상용소프트웨어인 Matlab과 LABVIEW를 통해 온라인 성능을 테스트 하고, 실제 교차로 상에서의 성능을 검증하였다.

주요어: 차량 인지 필터, 점데이터군 선처리 기법, 파티클 필터, 자율 주행 차량, 레이저 스캐너, 재귀 구조

학 번: 2015-20716